

-1-

Date: 11/12/01 Express Mail Label No. EL 930598088 US

Inventors: Michael J. Jones and Paul Viola

Attorney's Docket No.: 0918.2040-001

METHOD AND SYSTEM FOR OBJECT DETECTION IN DIGITAL IMAGES

RELATED APPLICATION

This application claims the benefit of U.S. Provisional Application No. 60/253,871, filed on November 29, 2000. The entire teachings of the above application
5 are incorporated herein by reference.

BACKGROUND OF THE INVENTION

Computers perform processing of graphic images in a digital format (e.g., photographs, still images from videos, and so on). Often, the goal of this processing is to locate objects of interest (e.g., faces) in an image. Given enough processing time to
10 process an image (typically using a digital processor), a computer is capable of detecting most or all well defined instances of an object in an image. One common goal for object detection is the detection of human faces, although computers can use object detection to detect various types of objects in an image. This process of detecting objects (e.g., faces) is useful for user interfaces, the scanning of image databases, in
15 teleconferencing, electronic processing of photographs, and other suitable areas. The appearance of objects varies greatly across individuals, images, camera locations, and illuminations.

There are a number of existing methods for detecting objects (e.g., faces) in images. Most existing, prior art approaches for detecting objects (e.g., faces) in an
20 image share a number of properties. First, for example, the conventional object detector

uses a learning algorithm based on a training data set that contains many examples of face and non-face image patches (the smallest possible region that might contain a face -- usually, a patch or subwindow 16x16 or 20x20 pixels). One such learning algorithm is based on conventional neural network approaches. In a learning phase based on a training data set, the learning algorithm constructs a classification function which can label patches as either face or non-face.

Finally, in a conventional approach, an object detector uses a scanning process to enumerate all possible patches (subwindows) within a given image. Each image contains many independent patches. Every unique location and scale in the image can yield an independent patch. In practice, a 320 pixel x 240 pixel image can produce approximately 50,000 patches (the number of patches scales quadratically with the scale of the image). The classification function is run against all such patches to detect the possible presence of an instance of the object in the patch. When an object detector, through one or more classification functions, detects an object (e.g, a face), the object detector records the location and scale of the patch for later output (e.g., reporting to an end-user of the computer).

To detect an object in a patch, many conventional, prior-art approaches work directly with intensity values (grayscale degree of lightness or darkness) of the pixels of the patches. In one prior art approach, the object detection software uses wavelet functions, such as Haar Basis functions that evaluate boxes in a patch, to detect an object in a patch.

SUMMARY OF THE INVENTION

In general, known prior art approaches may have a high rate of accurate detection of objects in an image, but may perform the object detection process in a relatively slow manner compared to the present invention. The classification may be based on gaussian discriminators, or on neural networks, which generally provide an accurate, but slow result. For example, to identify instances of an object in an image, prior art approaches typically take one to many seconds to process the image on a desktop personal computer (i.e., with an Intel 700 MHz Pentium III processor), whereas

a comparable computer configured according to the present invention takes substantially less than one second for the same image (approximately 1/15 of a second).

Almost all previous approaches to object detection perform a large amount of work (large number of computations by a digital processor) in the scanning process alone. In order to support scanning at multiple scales, the input image must be scaled down to multiple resolutions. In prior art approaches, this scaling is accomplished by constructing an image pyramid, that is, multiple copies of the original input image at different scales (e.g., each image is typically 80 % of the size of the previous image in the pyramid). This conventional process alone often requires 50-100 operations per pixel. One reason for this number of operations is that the computer must perform interpolation calculations between pixels as part of the scaling process. That is, if an initial 100 by 100 pixel image is scaled to a 80 by 80 pixel reduced image, then some pixels in the initial image must be eliminated, and interpolated values calculated for some pixels in the reduced image to reflect the elimination of pixels.

An object detector of the present invention engages in much less initial image processing compared to prior art approaches. The present invention creates and uses an image representation called an integral image, in contrast to typical prior art approaches that use an image pyramid. In a preferred embodiment, the present invention computes the integral image in less than about 10 operations per pixel. Nevertheless, the object detector of the present invention detects objects (e.g., faces) at any scale and location.

Prior art approaches evaluate pixels to identify an instance of an object in a subwindow, or rely on wavelets to identify a subwindow that is likely to contain an instance of an object. The present invention uses a feature representation, which detects objects (e.g., faces) by looking for the appearance of features, which have basic geometric shapes (e.g., based on rectangular boxes). These simple features, combined with the integral image, allow for a computationally efficient approach to identifying whether a given area in a subwindow (e.g., given boxes) has a feature of interest (that may identify an instance of the object, usually along with other features). This approach of the invention is more powerful and more efficient than looking at the pixels of the image itself as is done in many prior art approaches.

The present invention also provides a faster classification function than prior art classification functions. Using a cascaded approach, the present invention quickly determines if a face could potentially appear at a given scale and location. In some cases this can be done in 20 or less operations for a subwindow (i.e., patch) of the integral image. The present invention chains similar or homogenous types of classification functions together in a cascade of classification functions. This approach of the invention allows the object detector to discard quickly subwindows that do not show enough features of the object and to continue to process through the cascade only those subwindows that have enough features that indicate the likelihood of an instance of the object in the subwindow. Each classification function is a similar type of function. However, in sequence farther in position toward the end of the cascade, each classification function is increasingly accurate in detecting an instance of an object in a subwindow (e.g., relies on more features) .

The present invention can be used in real-time applications in which the appearance of an object can be used to drive a user interface. For example, an object detector for faces (i.e., face detector) that is designed in accordance with the present invention functions in a kiosk like those used in bank ATM's (automatic teller machines) or airport ticketing machines to identify faces in a real-time application. Knowledge of the location and scale of a face can be used in teleconferencing applications as well. For example, the camera can be made to zoom in on the faces of the participants. This ability to zoom enables increased quality and reduced bandwidth. A face detector based on the present invention can also play a central role in security camera applications. Such a face detector may be used to summarize many hours of airport surveillance tape into a single web page that shows a picture of each person that passed through a security checkpoint. Generally, the face detector of the present invention be used as the front end processor for a face recognition system.

The face detector can also be used in off-line applications such as image database browsing. Automatically determining which images contain faces yields important meta-information about each image. If users are searching for an image of a specific person, such meta-information can be used to insure that face recognition

system based on the present invention returns only images containing people. Detection of faces is also an important first step for the recognition of individuals.

Thus, the present invention provides computer apparatus and computer-implemented methods for detecting instances of objects in an image. In a preferred embodiment of the present invention, an object detection system includes an image scanner and an object detector. The image scanner places a working window at different positions in an input image such that the input image is divided into same dimension subwindows. The object detector provides a cascade of homogenous classification functions (classifiers). Each of the homogenous classification functions in sequence in the cascade respectively has increasing accuracy in identifying the certain objects. A homogenous classification function consists of a number of features. A feature is, for example, based on a set of eyes in a human face. A classifier (homogeneous classification function) detects that the presence of such a feature is likely in a subwindow by using two horizontal rectangular boxes, one overlaying the darker region indicated by the eyes, and a second horizontal rectangular box overlaying a lighter region indicated by the cheeks of the face. For each subwindow, the object detector employs the cascade of homogenous classification functions to detect instances of the certain objects in the image.

In one aspect of the present invention, the image scanner scales the dimensions of the subwindows by changing a size of the working window. The object detector scales the homogenous classification functions respectively for each different size of the working window. For each different size of the working window, (i) the image scanner repeats placing of the scaled working window at different positions in the input image to divide the input image into same dimension subwindows equal in size to the scaled working window, and (ii) the object detector repeats employing the cascade of scaled homogenous classification functions to detect the instances of the certain objects.

In another aspect, the object detection system includes an image integrator, which computes an integral image representation of the input image. The object detector uses the integral image representation in computing the homogenous classification functions.

In a further aspect the certain objects are human faces.

The object detection system includes, in a further aspect, a training server. The training server trains the homogenous classification functions in a learning phase based on a training data set and thereby identifies optimal such functions.

- 5 In another aspect, the training server constructs the cascade based on the optimal homogenous classification functions such that the object detector performs the employing of the cascade at an average processing rate of less than about 200 arithmetic operations for each subwindow.

- 10 In a further aspect, the processing rate is independent of the dimensions of the subwindows.

The object detector, in another aspect, provides to a computer output device an output image that identifies the detected instances of the certain objects based on the employing of the cascade.

- 15 In another aspect, each homogenous classification function (classifier) is based on one or more of the features and corresponding threshold functions. Each threshold function has a predefined feature threshold for the given feature indicating a presence of the given feature in the subwindow (or lack of such presence). Each homogeneous classification function is based on a summation function that sums all of the threshold functions for all of the features evaluated by the homogenous classification function.
- 20 Before summation, each threshold function is weighted by a predefined weight for that threshold function. The summation function includes a global threshold that determines whether or not a sum of the summation function indicates a detection of one of the instances of the certain object in the given subwindow.

- 25 In another aspect, the present invention provides a computer apparatus and methods for detecting objects in an input image at a relatively fast rate of processing. In one aspect, an object detection system includes an image scanner and an object detector. The image scanner divides an input image into subwindows. Each subwindow has a sufficient size to allow processing of features associated with the certain objects. The object detector processes the subwindows at an average processing rate of less than
- 30 about 200 arithmetic operations for each subwindow by (a) evaluating the features in

each subwindow, and (b) classifying each subwindow to detect an instance of the certain objects based on the evaluation of the features.

In another aspect, the processing rate is independent of dimensions of the subwindows.

- 5 The object detection system, in a further aspect, includes an image integrator. The image integrator computes an integral image representation of the input image. The object detector uses the integral image representation to compute homogenous classification functions for use in the processing of the subwindows.

- 10 In a further aspect, the object detector, for each subwindow, employs a cascade of optimal homogenous classification functions (classifiers). Each optimal homogenous classification function in sequence in the cascade respectively has increasing accuracy in identifying the features associated with the certain objects. At each optimal homogenous classification function in the cascade, (a) if a subject subwindow has the detected instance of the certain object, the object detector continues to pass the subject subwindow through the cascade for further processing, and (b) if the subject subwindow does not have the detected instance of the certain object, the object detector ceases to pass the subject subwindow through the cascade.

In another aspect, the certain objects are human faces.

BRIEF DESCRIPTION OF THE DRAWINGS

- 20 The foregoing and other objects, features and advantages of the invention will be apparent from the following more particular description of preferred embodiments of the invention, as illustrated in the accompanying drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention.

25 Fig. 1 is a block diagram of an object detection system according to the present invention.

Fig. 2 is a schematic diagram of the flow of control for the object detector of Fig. 1.

Fig. 3 is a pictorial illustration of rectangular features within windows according to the present invention.

Fig. 4 is a pictorial illustration of a rectangular feature of Fig. 3 overlaid on a representation of a face, such as found in input images.

5 Fig. 5 is a diagrammatic illustration of an integral image representation according to the present invention.

Fig. 6 is a diagrammatic illustration of a box car sum according to the present invention.

10 Fig. 7 is a schematic diagram of an object detection cascade according to the present invention.

Fig. 8 is a flow chart of a procedure of detecting instances of objects in an image according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

A description of preferred embodiments of the invention follows.

15 Fig. 1 is a block diagram of an object detection system 20 according to the present invention. The object detection system 20 includes a digital processor 24 and data storage 36 and is used to detect one or more instances of certain objects (i.e., predefined category of objects). The digital processor 24 hosts and executes an image integrator 26 and an object detector 28 in working memory. The input image 22 is a
20 digital image composed of bits (i.e., pixels) based on a photographic image, an image from a video, a computer created image, or other digital image. The output image 34 is a digital image composed of bits based on the input image 22 with highlighting that indicates the detected instances 38 of objects.

The input image 22 includes an object representation (i.e., instance) 38 of an
25 object displayed in a subwindow 42 of image 22. The object representation 38 is a recognizable instance of the object based on a realistic (e.g., photographic), drawn, painted, caricatured, cartoon, or other recognizable representation of the object. The contents of subwindow 42 is a part of the input image 22 based on a geometric shape (e.g., rectangle, square, or other shape). The object is a certain object based on a

predefined type or category of objects, such as faces, dogs, cars, trees, or other recognizable objects with a distinct appearance that distinguishes them from other objects.

The image integrator 26 is a software program, routine, object, module, or firmware or hardware entity that creates an integral image representation 44 (see Fig. 2) of the input image 22. The object detector 28 is a software program, routine, object, module, or firmware or hardware entity that detects instances of objects in an image 22 or part of an image 22 (e.g., patches or subwindows 42). The object detector 28 includes a classifier 30 (e.g., classification function based on one or more features of an object) that evaluates an image 22 or part of an image 22 to determine if an instance 38 of the object is detected or not. In a preferred embodiment, the object detector 28 uses a cascade of serially linked classifiers 30 (see Fig. 7). The object detector 28 includes an image scanner 32 that processes the input image 22 to divide the image 22 into smaller working pieces or subwindows 42.

The data storage 36 is a data storage device (e.g., one or more disk drives) associated with the object detection system 20 that stores data as needed for the digital processor 24, such as the input image 22, working copies of the input image 22 as it is processed, and the output image 34.

The training server 37 is a digital computer, such as a network server, that is used in the learning phase of the present invention to train the classifiers 30 (classification functions) based on a training data set that includes many digital images with predetermined or known instances of the object as well as negative example images showing what the object is not. The training server 37 functions to train the classifiers 30 in a preliminary or one-time learning phase. That is, the object detector 28 then uses the classifiers 30 to detect object representations 38 in images 22 without requiring further input from the training server 37.

In one embodiment, a computer program product 180, including a computer readable or usable medium (e.g., one or more CDROM's, diskettes, tapes, etc.), provides software instructions for the image integrator 26 and/or object detector 28. The computer program product 180 may be installed by any suitable software installation

procedure, as is well known in the art. In another embodiment, the software instructions may also be downloaded over an appropriate connection. A computer program propagated signal product 182 embodied on a propagated signal on a propagation medium (e.g., a radio wave, an infrared wave, a laser wave, a sound wave, or an electrical wave propagated over the Internet or other network) provides software instructions for the image integrator 26 and/or object detector 28. In alternate embodiments, the propagated signal is an analog carrier wave or digital signal carried on the propagated medium. For example, the propagated signal may be a digitized signal propagated over the Internet or other network. In one embodiment, the propagated signal is a signal that is transmitted over the propagation medium over a period of time, such as the instructions for a software application sent in packets over a network over a period of milliseconds, seconds, minutes, or longer. In another embodiment, the computer readable medium of the computer program product 180 is a propagation medium that the computer may receive and read, such as by receiving the propagation medium and identifying a propagated signal embodied in the propagation medium, as described above for the computer program propagated signal product 182.

Fig. 2 is a schematic diagram of the flow of control for the object detector 28 of Fig. 1. The object detection system 20 receives the input image 22, which may be received over a network, from a photographic or video camera, from data storage 36, or from another suitable source of digital images. The image integrator 26 computes the integral image 44, which is a representation of the input image 22 as described in more detail later in Fig. 5. The image scanner 32 then scans the integral image 44 to divide the integral image 44 into subwindows 42 and uses the classification function or classifier 30 to classify each subwindow 42 in the integral image 44 as having detected faces 46 or not. The object detector 28 then outputs an output image 34 with the object representations 38 highlighted (e.g., with bounding boxes surrounding each detected face 46).

Fig. 3 is a pictorial illustration of rectangular features 54 (e.g., 54A, 54B, 54C, 54D) within windows (or subwindows) 42 (e.g., 42A, 42B, 42C, 42D) of an image 22 according to the present invention. Fig. 3 illustrates four windows 42A, 42B, 42C, and

42D, that represent subwindows in one or more images 22. Within the windows 42A, 42B, 42C, and 42D are respective rectangular features 54A, 54B, 54C, and 54D.

Rectangular feature 54A is composed of two vertical rectangular boxes, 54A-1 and 54A-2. The box 54A-2 is shaded to indicate that, when feature 54A has a “true”

5 threshold value, box 54A-2 overlays a darker region of the image 22 than the region overlaid by box 54A-1. See the discussion for Fig. 4, for more details on the threshold function for a feature 54. Rectangular feature 54B is composed of two horizontal boxes 54B-1 and 54B-2 in rectangular feature 54B. The uppermost box 54B-1 is shaded to indicate that it is a darker box when feature 54B has a “true” threshold value.

10 Rectangular feature 54C is composed of three boxes, 54C-1, 54C-2, and 54C-3. In rectangular box 54C, the middle box 54C-2 is shaded to indicate that it overlays a darker region of an object representation 38 when feature 54C has a “true” threshold value. Rectangular feature 54D is composed of four rectangular boxes, 54D-1, 54D-2, 54D-3, and 54D-4. In rectangular feature 54D, the boxes 54D-2 and 54D-4 are both
15 shaded to indicate regions that are darker than the other boxes, 54D-1 and 54D-3, in the rectangular feature 54D when the feature 54D has a “true” threshold value.

The evaluation function f for each rectangular feature 54A, 54B, 54C, 54D determines the value of the feature 54A, 54B, 54C, and 54D by subtracting the sum of the pixels (intensity values) which lie within the white rectangles from the sum of the
20 pixels in the darker rectangles. For example, the evaluation function f for feature 54A determines the value by subtracting the sum of the pixels which lie within the white rectangle 54A-1 from the sum of pixels in the darker rectangle 54A-2. Another possible feature has the sum for 54A-2 subtracted from the sum for 54A-1.

The evaluation function f for the three rectangle feature 54C computes the sum
25 within two outside rectangles 54C-1 and 54C-3, and subtracts from the sum of the pixels in the center rectangle, 54C-2 to obtain the value for the feature 54C.

The four features 54 shown in Fig. 3 are examples of the features 54 used to build a classifier. The full set of features 54 includes variations of these examples in terms of their position, size and orientation (vertical or horizontal).

30 In general, the classification function or classifier 30 (Fig. 1) uses one or more

rectangular features 54 to detect simple features in an object. For example, the classifier 30 uses the four rectangle feature 54D to detect a diagonal pattern or line within an object representation 38. The classifier 30 uses the three rectangle feature 54C to detect for example, a mouth in a face, because the darkened region in 54C-2, as shown in Fig. 5 3, would overlay or represent the mouth and the lighter rectangles, 54C-1 and 54C-3, would overlay or represent the cheeks of the face.

This rectangular feature approach of the present invention uses rectangular features 54, rather than evaluating the pixels of the image 22 directly. The feature-based system of the present invention typically operates more quickly than a traditional pixel- 10 based system because the rectangle features 54 can be computed very quickly (without looking at every pixel in the feature 54) using the integral image representation 44.

Fig. 4 is a pictorial illustration of a rectangular feature 54B overlaid on a representation 38 of a face image according to the present invention. This rectangular feature 54B measures the difference in intensity between the region of the eyes (darker 15 region) and a region across the upper cheeks (lighter region). This feature 54B capitalizes on the observation that the eye region is often darker than the cheeks. Thus, when the feature 54B is overlaid on the facial representation 38, as shown in Fig. 4, the sum of the pixels (intensity or gray scale values) in the upper horizontal box 54B-1 indicate a darker region (the eyes) than the sum of the pixels in the lower horizontal box 20 54B-2, which indicates a lighter region (the cheeks). In a preferred embodiment, the evaluation function f for feature 54B subtracts the sum of the pixels in the lower horizontal box 54B-2 from the sum of the pixels in the upper horizontal box 54B-1. A threshold function h for feature 54B then compares this difference to a threshold to determine a true value (indicating a high likelihood that the feature 54B is overlaying a 25 set of eyes in the subwindow 42) or a false value (indicating a low likelihood that the feature 54B is overlaying a set of eyes in the subwindow 42). In other words, in a preferred embodiment, if the difference is below the threshold value, then the threshold function h for feature 54B has a false value (e.g., 0), and if the difference is above the threshold value, then the threshold function h for feature 54B has a true value (e.g., plus 30 1). In another embodiment, this threshold relationship is reversed. If the difference is

below the threshold value, then the threshold function h for feature 54B has a true value, and if the difference is above the threshold value, then the threshold function h has a false value.

In general, the form of the classifier 30 (classification function) is a collection of features 54, a set of weights and a final threshold theta (global threshold for the classifier 30 as a whole). Each feature 54 is given a unique weight and the weighted sum of features 54 is computed. If this weighted sum is above the global threshold, the classifier 30 returns TRUE. The true value indicates that the classifier 30 has found that the subject window 42 has a high likelihood of having an object representation 38 within the window 42.

In mathematical terms, in a preferred embodiment, the threshold function h (also termed a “weak classifier”) is expressed as the following:

$$h_j = \begin{cases} 1, & \text{if } p_j f_j(x) > p_j T_j \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where x is a 24 pixel x 24 pixel patch (subwindow) 42 of the input image 22. The variable x contains the values in the patch 42 an array of numbers that is stored in a linear manner so that x is a vector $[0 \dots 576]$ of one byte integers (each integer having a value of 0-255 representing a gray scale intensity value for one pixel in the image 22).

In general, without reference to the subscript j , (which will be discussed next) the function f is the evaluation function for a rectangular feature 54, as discussed for Fig. 3. In general, the value T is the threshold, and the evaluation function f must evaluate to a value larger than T for the threshold function h to have a value of 1. A polarity value p (having a value of +1 or -1) indicates the direction of the inequality sign in equation (1). Thus, if the polarity value is positive (+1), then the value of f must evaluate to a value larger than the threshold T for the threshold function h to have a value of 1, and if the polarity value is negative (-1), then the value of f must evaluate to a value less than the threshold T for the threshold function h to have a value of 1.

The subscript j is applicable when the classifier 30 (classification function) includes more than one feature 54, and thus more than one evaluation function f_j (one

for each feature), and more than one threshold function h_j (one for each feature). Each feature has its own threshold value T_j .

If the classifier 30 (classification function) has more than one feature 54, then the classifier 30 is based on the summation of the threshold functions h , as follows:

$$5 \quad \sum_{j=1}^N w_j h_j(x) > \theta \quad (2)$$

The function h is the threshold function as defined in equation (1). The weight w is the weight applied to each threshold function h that represents the weight or importance given to the feature 54 for that threshold function. In other words, the weight w measures how important that feature 54 is in identifying an instance 38 of the object in the working patch or subwindow 42. The threshold θ is a global threshold for the summation as a whole. In other words, if the sum indicated in equation (2) is greater than this threshold, then the classifier 30 has a value of TRUE, indicating that the subwindow 42 is likely to contain an instance of the object. If the sum indicated in equation (2) is less than this global threshold, then the classifier 30 has a value of FALSE, indicating that the subwindow 42 is not likely to contain an instance of the object. The weight w and global threshold θ are determined in a learning phase based on a training data set, to be discussed in more detail later. The value N is the number of features 54 that are evaluated in the classifier 30. If N is large, then the classifier 30 is more accurate, and if N is small, then the classifier 30 is faster.

Fig. 5 is a diagrammatic illustration of an integral image 44 according to the present invention. The (x,y) value of the integral image 44 at point 62 in a preferred embodiment, is the sum of the pixels (intensity values) in an integral area 60, that is, the sum of all the pixels above and to the left of the point 62. The location of point 62 in Fig. 5 is shown as an example only, and point 62 may represent any point in the integral image 44 (except for point 64, to be discussed next). Each pixel has an intensity value, which is a grayscale value based on a scale of 0 to 255. To create the integral image 44, first the integrator 26 retains the pixel value (e.g., intensity value) of the point 64 at the upper left hand corner of the integral image 44. Then the integrator 26 moves to each point 62 in the integral image 44 and calculates the integral value for that point 62 by

summing the pixel values for all of the pixels above and to the left of the subject point, as described above. In a preferred embodiment, the integrator 26 performs these calculations in one pass over all of the points 62 in the input image 22 to create the integral image 44. Moving left to right, then top to bottom, the integrator 26 keeps a running sum for each row.

Fig. 6 is a diagrammatic illustration of a rectangular (also termed box car) sum according to the present invention. Fig. 6 illustrates rectangular regions or boxes 70, 72, 74, and 76 and points p1, p2, p3, and p4 in the integral image 44. Each point, p1, p2, p3, or p4, has a value in the integral image 44 as described above for points 62 in Fig. 5.

The integral image 44 contains all the information necessary to compute a box car sum of the original image 22 in constant time. The box car sum is the sum of the pixels inside of the rectangular region denoted by its boundary points (x1, y1, x2, y2). The evaluation function f for a rectangular feature 54 computes the sum over a large box (e.g., box 70) without any additional computational effort than computing the sum over a small box (e.g., 76).

Point p1 contains (i.e., the value of p1 is) the sum of the pixels in box 70. Point p2 contains the sum of the pixels in boxes 70 and 72. Point p3 contains the sum of the pixels in boxes 70 and 74. Finally, point p4 contains the sum of the pixels in boxes 70, 72, 74 and 76. If the evaluation function f for a feature 54 is computing the sum of the pixels just in box 76, the evaluation function f only needs to compute $p1 - p2 - p3 + p4$.

The classifier 30 uses the integral image 44 and rectangle sums to evaluate, in a preferred embodiment, the features 54. In a preferred embodiment, the object detector 28 uses a cascade of such classifiers 30, as will be described in more detail later (see Fig. 7).

The classifier 30 (classification function) is designed to operate using rectangular features 54 (also termed box car averages) for two reasons. One reason is that a rectangular feature 54 contains more information and is more robust than evaluating the values of single pixels. The second reason is that by using the integral image representation 44, rectangular features 54 can be computed at any scale in constant time. In other words, larger rectangular features 54 take the same time to

compute as smaller ones. Hence, no additional processing is necessary to evaluate a classifier 30 at any scale (i.e., any size subwindow 42). In other words, rather than computing many scaled versions of the original image 22 (i.e., construct an image pyramid as is done with prior art approaches) which is then divided into 16 pixel x16 pixel (or other size) patches, the approach of the present invention uses the same integral image 44 for computation at each scale. Object detection at different scales is achieved by scaling the classifier 30 itself (see below).

In a preferred embodiment, a feature 54 has weights associated with each box in the feature 54 so that the sum of the weights in a feature 54 is zero. For example, each three rectangle (triplet) feature (e.g., 54C in Fig. 3) is computed as the weighted sum (-1, 2, -1) of three identical, adjacent and non-overlapping rectangular sums arranged either vertically or horizontally. For the triplet feature 54C in Fig. 3, the sum of pixels in box 54C-1 has a weight of -1; the sum of pixels in box 54C-2 has a weight of 2; and the sum of pixels in box 54C-3 has a weight of -1.

The set of all possible features 54 (from which any given classifier 30 will use only a few) is the set of all features 54 which will fit, in a preferred embodiment, within a 16 pixel x16 pixel patch (there are approximately 80,000 such features 54). Note that all of these operations can be computed using fixed point (integer) representations. This allows for implementations on very simple microprocessors and gate arrays.

Given the rectangular feature 54 representation, the image scanner 32 scans the classifier 30 across locations in the integral image 44 and scales by shifting and scaling the boundary points of the features 54. In other words, the object detector 28 precomputes the classifier 30 (classification function) for each scale, and then the scanner 32 need only scan the appropriately scaled classifier 30 for the given scale across locations in the integral image 44 at run time. In a preferred embodiment, the object detector 28 scales the size of each feature 54 in a classifier 30 proportionately to the desired scale. Also, the object detector 28 scales the value of the threshold T for each feature 54 accordingly to correspond to the change in scale. Thus, the object detector 28 creates a scaled classifier 30 by scaling all of the features 54 and adjusting the thresholds T for each feature 54 appropriately.

Fig. 7 is a schematic diagram of a cascade 80 of classifiers 30-1, 30-2, 30-3, 30-4 (classification functions) for a preferred embodiment of the invention. The object detector 28 uses the cascade 80 to detect one or more instances 38 of an object in an image 22. The cascade 80 applies the series of classifiers 30-1, 30-2, 30-3, 30-4 to every subwindow 42 provided in the initial set of subwindows 82, which represents all of the subwindows 42 in the image 22 as determined by the image scanner 32. The initial classifier 30-1 eliminates a large number of negative examples (e.g., subwindows 42 without instances 38 of an object) with very little processing. The initial classifier 30-1 utilizes a limited number of features 54 (e.g., one to five features 54) to identify a large percentage, such as 50%, of the initial set of subwindows 82 which are unlikely to have any instances 38 of the object. For example, if the initial image 22 is a photograph with a large area representing a white wall, then the initial classifier 30-1 eliminates the subwindows 42 that cover the area of that white wall. In Fig. 7, this elimination process 88 is indicated by the letter "F" for "False", and the rejected set of subwindows 84 indicates those subwindows 42 eliminated by the classifiers 30-1, 30-2, 30-3, and 30-4, respectively. The retention 90 of windows is indicated by the letter "T" for "True", as the classifiers 30-1, 30-2, 30-3, and 30-4 pass on subwindows 42 that are not eliminated. After the classifier 30-4, the retained subwindows 42 are passed on for further processing 86, such as processing by additional classifiers 30 in the cascade 80 or for output processing such as highlighting the instances 38 of objects in an output image 34 for viewing by an end-user of the object detection system 20.

The later classifiers 30-2, 30-3, and 30-4 eliminate additional negatives (e.g., rejected subwindows 84) but require additional computation than what was required for the initial classifier 30-1. The initial classifier 30-1 removes rejected subwindows 84 that are easiest to reject, such as those with no distinguishing features 54 in the subwindows 84. At later stages, the classifiers 30-2, 30-3, and 30-4 require more work, as indicated by evaluating progressively larger numbers of features 54 in each classifier 30-2, 30-3, and 30-4.

For example, the first classifier 30-1 evaluates one or two features 54 and may eliminate one half of the initial set of subwindows 82. Eliminating one-half of the

negative examples applies to the example of faces and may not hold for other objects.

The second classifier 30-2 evaluates five features 54 and eliminates one half of the subwindows 42 that the second classifier 30-2 receives from the first classifier 30-1.

The proportion of subwindows 42 that are retained is referred to as the false positive

5 rate (e.g., one-half). Thus, the third classifier 30-3 receives one quarter (the false positive rate) of the initial set of subwindows 82. The third classifier 30-3 evaluates 10 features 54 and eliminates further rejected subwindows 84. By the end of a cascade 80 the last classifier 30 (e.g., 30-4) may process 100 features 54 to eliminate one-half of the retained subwindows 42 that the last classifier 30 has received. However, the
10 processing by the last classifier 30 does not take a large amount of computing time (e.g., compared to what would be required to process all of the subwindows 82), because the last classifier 30 receives a relatively small number of retained subwindows 42 out of the large set of all subwindows 82 received by the initial classifier 30-1.

In a preferred embodiment, the cascade has about 35 classifiers 30 and the last
15 classifier 30 in the cascade 80 processes about 200 to 300 features 54. In other words, as more and more windows 42 are eliminated to the set of rejected subwindows 84, the classifier 30 in the later stages (e.g., 30-2, 30-3, and 30-4) must process more features 54 to distinguish the instances 38 of an object (e.g., a human face) from other objects (e.g., a dog's face) and patterns in general (e.g., the folds of cloth in clothing).

20 In one example, the cascade 80 includes six classifiers 30 with 2, 20, 100, 200, 250, and 400 features 54 respectively. This cascade 80 is significantly faster than a 20 feature 54 classifier 30 running alone, but this cascade 80 has a much better classification performance (i.e., much more accurate detection of instances 38 of objects).

25 Each classifier 30 is based on the equations (1) and (2) given above, but with differing values for the number, N , of features 54 evaluated, the weights, w_j , feature thresholds, T_j , and the global threshold, θ . Thus, in a preferred embodiment, all of the classifiers 30 in a cascade 80 are homogenous (that is, homogeneous classification functions). That is, they are made up of the same types of functions and equations as
30 indicated by equations (1) and (2).

The techniques of the present invention use a learning phase to determine the differing values for the number, N , of features 54 evaluated, the weights, w_j , feature thresholds, T_j , and the global threshold, θ , for each classifier 30. Each classifier 30 is trained based on a training data set. Typically, the training data set is a large number of different photographs with varying instances 38 of the object. In one embodiment, the learning phase is based on perceptron principles. In a preferred embodiment, a training server 37 performs the learning phase and provides the classifiers 30 to be used by the object detector 28. But after the learning phase, the training server 37 does not participate in the processing of the input images 22.

In a preferred embodiment, the learning phase for the initial training of each classifier 30 (classification function) is based on the AdaBoost learning procedure. For example, see Freund & Schapire, "Experiments with a New Boosting Algorithm," *Machine Learning: Proceedings of the Thirteenth International Conference*, 1996.

In a preferred embodiment, the AdaBoost approach is used both to select a small set of features 54 for each classifier 30 and to perform the initial training of the classifier 30. The learning phase uses the AdaBoost learning procedure to determine the number, N , of features 54 evaluated, the weights, w_j , feature thresholds, T_j , and the global threshold, θ for each classifier 30, thus producing optimal homogeneous classification functions 30.

The AdaBoost learning procedure selects from among the potential features 54 available in a subwindow 42. The AdaBoost learning procedure insures (under reasonable conditions) that the training error will eventually go to zero, and that the generalization error on a test set (used for training in the learning phase) will be reduced as the number of features 54 (in a classifier 30) is increased.

The AdaBoost learning technique is used to boost the classification performance of a simple learning algorithm (e.g., it might be used to boost the performance of a simple perceptron). AdaBoost does this by combining a collection of weak classification functions to form a strong classification function or classifier (a classification function or classifier 30 as described herein). As described above for equation (1), the threshold function, h_j , is considered a weak classification function

(e.g., that represents a weak classifier based on one feature 54 only). The AdaBoost approach determines the optimal threshold function, h_j , for each feature 54, such that the minimum number of examples are misclassified.

The AdaBoost approach also determines the classification function or classifier 30 represented by equation (2), which represents the sum of weak classification functions (e.g., weak classifiers or threshold functions) h_j , determining the values for the weights, w_j , and the global threshold θ . See the Appendix for more information on the AdaBoost learning technique in the preferred embodiment of the invention.

In general, the cascade 80 is constructed by initially training classifiers 30 using AdaBoost which provides an initial set of default global thresholds θ .

Then, in a further training phase of the classifiers 30 after the initial training of the classifiers 30, the techniques of the present invention adjust the global threshold θ for each classifier 30 to minimize false negatives (that is, rejecting subwindows 42 that do have instances 38 of the object in them). The default global threshold θ is designed to yield a low error rate on the training data. In general, a lower global threshold θ yields higher detection rates and higher false positive rates.

Using the present invention, the goal is, for each stage of the cascade 80, to obtain a high detection rate (i.e., few or no false negatives) and a minimal false positive rate. For example, a first stage classifier (30-1 in Fig. 7) can be constructed from a two- feature strong classifier 30 by reducing the global threshold, θ , to minimize false negatives. Measured against a validation training set, the global threshold, θ , can be adjusted to detect 100% of the instances 38 of objects with a false positive rate of 40% for the example of detecting faces.

In the approach of the present invention, each stage (i.e., classifier 30) in the cascade 80 reduces the false positive rate significantly while not lowering or lowering only slightly the detection rate. A target is selected (e.g., by a designer or programmer of the object detection system 20) for the minimum reduction in false positives and the maximum acceptable decrease in detection rate. The learning phase trains each stage (classifier 30) by adding features 54 until the target detection and false positive rates are met (these rates are determined by testing on a validation set). Stages (classifiers 30)

are added to the cascade 80 until the overall target for false positive and detection rates are met. The correct detection rate (of subwindows 42 with instances 38 of objects) can be compared to the false positive rate by plotting the correct detection rate against the false positive rate in a ROC (receiver operating characteristic) curve (see Appendix).

5 Fig. 8 is a flow chart of a procedure summarizing the steps for detecting instances 38 of objects in an image 22 according to the present invention. In step 102, the image integrator 26 receives the input image 22 and computes an integral image 44 based on the input image 22. The image integrator 26 computes the integral image 44 using the procedures or approach described for Fig. 5.

10 In step 104, the image scanner 32 divides the integral image 44 into subwindows 42, having identical dimensions. The image scanner 32 performs this division by placing a working window at different positions in the input image 22 such that the image 22 is divided into a number of subwindows 42 having the same dimensions.

At step 106, the object detector 28 evaluates the subwindows 42 by using a
15 cascade 80 of homogeneous classification functions or classifiers 30, as described for Fig. 7. As described previously, the cascade 80 is a series of optimal homogeneous classification functions 30. Each of the homogeneous classification functions 30, in sequence in the cascade 80, has increasing accuracy in identifying features 54 associated with the objects. The classification function 30 evaluates the features 54 in each
20 subwindow 42. The classification function 30 has been optimized in a learning phase based on a training data set as described previously, so that each classification function 30 is optimized to detect instances 38 of objects based on the features 54, depending on which stage of the cascade 80 the classification function 30 represents. The
classification function 30 is optimized to be accurate, that is to detect instances 38 of
25 objects in the subwindows 42, based on a number of features 54 for that classification function 30.

In step 108, the classification function 30 detects instances 38 of objects in the subwindows 42 based on the evaluation performed in step 106. Thus, the classification function 30 classifies each subwindow 42 as having an instance 38 of an object or as not
30 having an instance 38 of an object. The procedure described so far in steps 102 through

108 describes how the object classification function 30 and cascade 80 perform based on one scale of detecting instances 38 of objects, that is detecting those instances 38 in one scale or one size of subwindows 42 that are based on dividing the image 22 into those same sized subwindows 42.

5 Thus, in step 110 (which may be performed earlier, such as before step 102) the object detector 28 scales the classification function 30 to different sizes and repeats the process of scaling, that is, based on a predefined number of scaling levels. Thus, the classification function 30 is in fact, increased in size for the next scaling level. This increase in size of the classification function 30 is accomplished by increasing the size
10 of the features 54 that are part of the classification function 30 and also adjusting the feature threshold h appropriately for the adjustment in the size of scale. Thus, for any given increase in scale, which reflects an increase in the size of the subwindows 42, all of the classification functions 30 in the cascade 80 scale in a corresponding manner to the increasing scale to the subwindow size 42. In a preferred embodiment, the scaling
15 of the classifiers (classification functions) 30 is done before scanning the image so that the classification functions 30 are already scaled to a certain number of scales, such as eight to ten different scaling levels in a preferred embodiment.

 In step 112, the object detector 28 determines if the image 22 has been processed at each scaling level (for a predefined number of scaling levels). If not so, the object
20 detector 28 repeats the process (steps 104-108) by returning to step 104. Then the procedure 100 returns to step 104 to repeat the process of dividing the image 22 into subwindows 42, having identical dimensions at the new scaled size of the subwindows 42. Then for that same scale, the procedure 100 proceeds through steps 106 and 108, as described above.

25 While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of the invention encompassed by the appended claims.

 For example, the procedures of the present invention described herein could be
30 partially or fully implemented in hardware such as a field programmable gate array

(FPGA). The integral image representation 44 is computable with a few adder blocks. Detection at any given scale could be computed by streaming the integral image 44 through a FIFO (First In, First Out). Each feature 54 is then implemented using an ALU (Arithmetic Logic Unit) block.

- 5 In another example, features 54 may be composed of boxes that do not contact each other, so that a two rectangle feature 54 may include two independent boxes 42 at different parts of a feature (e.g., the top and the bottom of a subwindow 42).

- In a further example, the object detection system 20 is part of desktop computer, personal computer, laptop computer, or other computing system. The object detection
10 system 20 may also be part of a larger system, such as a security system for a company or other facility, such as an airport.